

ARCHITETTURA DEI SISTEMI ELETTRONICI

LEZIONE N° 5

- **Codici numerici**
- **Codici alfanumerici**
- **Riepilogo operazioni**
- **Virgola mobile**

A.S.E.

5.1

Richiami

- **Conversioni**
- **Aritmetica binaria**
- **Rappresentazione di numeri con segno**
- **Addizione in C2**

A.S.E.

5.2

CODICI

- Numeri binari OK per sistemi elettronici digitali
- Numeri decimali OK per sistema “uomo”
- Necessità di rappresentare anche non numeri
- Codifica binaria di informazioni varie
- Esempio
 - Codifica binaria di numeri decimali

A.S.E.

5.3

BCD (Binary-Coded Decimal numbers)

- Necessità di rappresentare i numeri decimali in codice binario
- 8421 BCD
- si codifica in binario ciascuna cifra decimale utilizzando i primi 10 numeri binari su 4 bit
- Esempio
- 453_{10}

4	5	3
0100	0101	0011
- 010001010011
- è possibile eseguire somme e sottrazioni in BCD

A.S.E.

5.4

Altri codici (1)

- Codici decimali pesati

B10	C9	8421	2421	5421	5043210
0	9	0000	0000	0000	0100001
1	8	0001	0001	0001	0100010
2	7	0010	0010	0010	0100100
3	6	0011	0011	0011	0101000
4	5	0100	0100	0100	0110000
5	4	0101	1011	1000	1000001
6	3	0110	1100	1001	1000010
7	2	0111	1101	1010	1000100
8	1	1000	1110	1011	1001000
9	0	1001	1111	1100	1010000

A.S.E.

5.5

Altri codici (2)

- Codici decimali non pesati

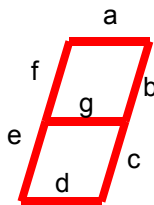
B10	Eccesso a 3	2 su 5
0	0011	11000
1	0100	00011
2	0101	00101
3	0110	00110
4	0111	01001
5	1000	01010
6	1001	01100
7	1010	10001
8	1011	10010
9	1100	10100

A.S.E.

5.6

BCD – Sette Segmenti

- Per visualizzare le cifre decimali si usa frequentemente un Display a sette segmenti



- È possibile realizzare un codificatore
 - BCD SETTE SEGMENTI

A.S.E.

5.7

Tabella di “Corrispondenze”

- La tabella risulta

	8	4	2	1	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	1	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

A.S.E.

5.8

Codice Gray

- **Codici a distanza unitaria**

- La codifica di n e n+1 differiscono sempre di un solo bit
- Codice inverso

1	2	3
0	0 0	0 0 0
1	0 1	0 0 1
	1 1	0 1 1
	1 0	0 1 0
		1 1 1
		1 0 1
		1 0 0

A.S.E.

5.9

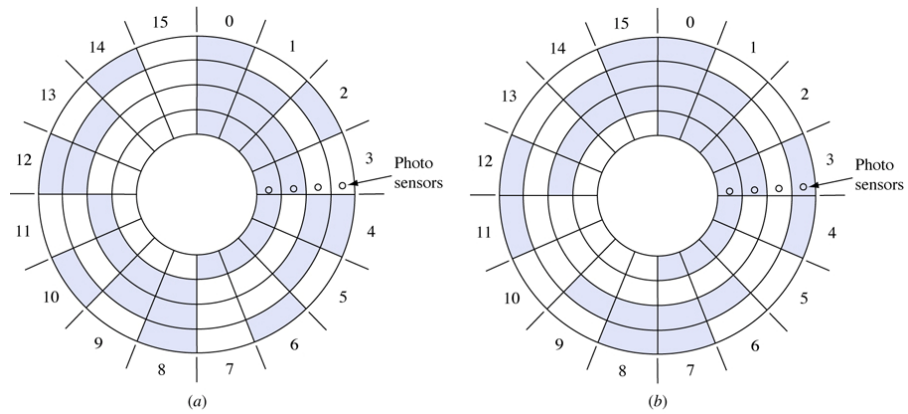
Codice Gray a 4 bit

Dec	ExD	Binario				Gray				
0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	1	1
2	2	0	0	1	0	0	0	1	1	3
3	3	0	0	1	1	0	0	1	0	2
4	4	0	1	0	0	0	1	1	0	6
5	5	0	1	0	1	0	1	1	1	7
6	6	0	1	1	0	0	1	0	1	5
7	7	0	1	1	1	0	1	0	0	4
8	8	1	0	0	0	1	1	0	0	12
9	9	1	0	0	1	1	1	0	1	13
10	A	1	0	1	0	1	1	1	1	15
11	B	1	0	1	1	1	1	1	0	14
12	C	1	1	0	0	1	0	1	0	10
13	D	1	1	0	1	1	0	1	1	11
14	E	1	1	1	0	1	0	0	1	9
15	F	1	1	1	1	1	0	0	0	8

A.S.E.

5.10

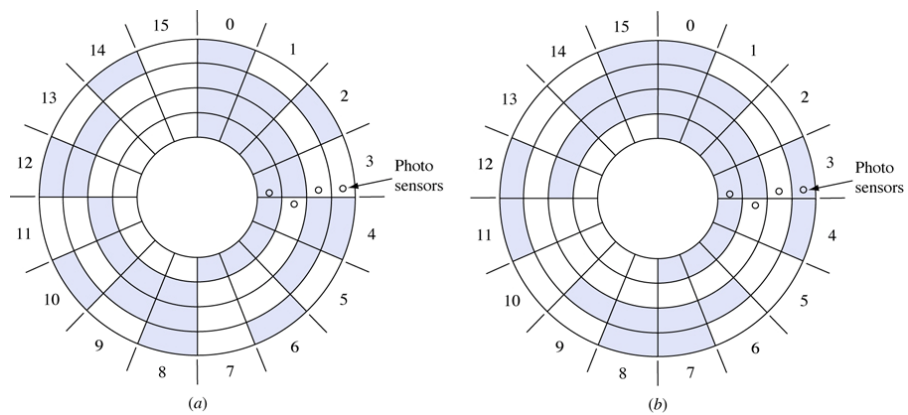
ENCODER 1



A.S.E.

5.11

ENCODER 2



A.S.E.

5.12

Codici alfanumerici

- **Necessità di rappresentare caratteri alfabetici con un codice binario**
- **Alfabeto = 26 simboli diversi**
- **Necessità di maiuscole e minuscole**
- **Numeri = 10 simboli**
- **Caratteri speciali**
- **Codice ASCII a 128 simboli**
- **UNICODE 16 bit** **simboli e ideogrammi (universale)**

A.S.E.

5.13

Codice ASCII

				$b_6 b_5 b_4$							
b_3	b_2	b_1	b_0	000	001	010	011	100	101	110	111
0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	"	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	\$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB	'	7	G	W	g	w
1	0	0	0	BS	CAN	(8	H	X	h	x
1	0	0	1	HT	EM)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[k	{
1	1	0	0	FF	FS	,	<	L	\	l	
1	1	0	1	CR	GS	-	=	M]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	—	o	DEL

A.S.E.

5.14

Codice ASCII caratteri di controllo

Control Characters			
NUL	Null	DC1	Device Control 1
SOH	Start of Heading	DC2	Device Control 2
STX	Start of Text	DC3	Device Control 3
ETX	End of Text	DC4	Device Control 4
EOT	End of Transmission	NAK	Negative Acknowledge
ENQ	Enquiry	SYN	Synchronous Idle
ACK	Acknowledge	ETB	End of Transmission Block
BEL	Bell	CAN	Cancel
BS	Backspace	EM	End of Medium
HT	Horizontal Tab	SUB	Substitute
LF	Line Feed	ESC	Escape
VT	Vertical Tab	FS	File Separator
FF	Form Feed	GS	Group Separator
CR	Carriage Return	RS	Record Separator
SO	Shift Out	US	Unit Separator
SI	Shift In	SP	Space
DLE	Data Link Escape	DEL	Delete

A.S.E.

5.15

Riconoscimento d'errore

- **Errore di trasmissione a distanza (Disturbi)**
- **Stringa digitale di “0” e “1”**
- **L'errore si manifesta nel convertire uno 0 in 1 o viceversa**
- **Su una parola di “K” bit la probabilità che ci siano due errori è molto bassa**
- **Codici a ridondanza (già visti “5043210” e due su cinque)**
- **Esempio**
 - **Numero 7 => 1000100 ricevuto 1010100**

A.S.E.

5.16

Bit di parità

- **Necessità di individuare eventuali errori di trasmissione**
- **Si aggiunge un bit (rappresentazione su 8 bit)**
- **Il numero complessivo di “1” è sempre pari**

Simbolo	Codice ASCII	Parità PARI	Parità DISPARI
T	1010100	11010100	01010100
7	0110111	10110111	00110111
-	0101101	00101101	10101101

A.S.E.

5.17

Riepilogo

-

A.S.E.

5.18

Interi Assoluti

- Base 2
- Dinamica
- dati "N" bit

$$0 \leq W \leq 2^N - 1$$

- Esempio N = 8

Base 10	Base 2
0	00000000
16	00010000
95	01011111
255	11111111

A.S.E.

5.19

Somma di Interi Assoluti

- N = 8

Base 10	Base 2
	01110110
43+	00101011
25=	00011001
68 .	01000100
	100011110
139+	10001011
141=	10001101
280.	100011000

Il carry N+1 indica l'overflow

24

La somma di due numeri di N bit è rappresentabile sempre su N + 1 bit

A.S.E.

5.20

Sottrazione di Interi Assoluti

- $N = 8$

Base 10	Base 2
	<u>11110000</u>
143-	10001111
86=	<u>01010110</u>
57.	00111001
	<u>10000000</u>
95-	01011111
141=	<u>10001101</u>
- 45.	<u>11010010</u>

Il borrow N+1 indica l'errore

210

A.S.E.

5.21

Prodotto di Interi Assoluti

- $N = 5$

Il prodotto di due numeri su N bit è rappresentabile su 2N bit

Base 10	Base 2
19 x	10011
<u>23 =</u>	<u>10111</u>
57.	10011
380.	100110
<u>437.</u>	1001100
	0000000
	<u>10011000</u>
	0110110101

$$(2^N - 1) \cdot (2^N - 1) = 2^{2N} - 2^{(N+1)} + 1 < 2^{2N} - 1$$

A.S.E.

5.22

Interi Relativi

- **Complemento a 2**
- **Dinamica**
- **dati “N” bit** $-2^{N-1} \leq W' \leq 2^{N-1} - 1$

$$W' = \left| 2^N + W \right|_{2^N}$$

- **Esempio N = 8 (-128 < W < 127)**

Base 10		Base 2 C-2
87	256+87=343	01010111
-123	256-123=133	10000101

A.S.E.

5.23

Complemento a 2

- **Primo metodo**
 - **Applicare la definizione**

$$W' = \left| 2^N + W \right|_{2^N}$$

- **Secondo metodo**
 - **complemento bit a bit più 1**

$$+87 = 01010111$$

$$-87 = 10101000 + 1 = 10101001$$

A.S.E.

5.24

Somma di Interi Relativi (C-2)

- Se non c'è overflow la somma è sempre corretta
 - W' rappresentazione di W in C-2
 - Z' rappresentazione di Z in C-2

$$W' + Z' = \left\| 2^N + W \right\|_{2^N} + \left\| 2^N + Z \right\|_{2^N} = \left\| 2^N + W + Z \right\|_{2^N}$$

Base 10	Base 2	Base 10	Base 2
	<u>01110110</u>		<u>111011110</u>
43+	00101011	43+	00101011
25=	<u>00011001</u>	-25=	<u>11100111</u>
<u>68.</u>	01000100	<u>18.</u>	00010010

A.S.E.

5.25

Sottrazione di Interi Relativi (C-2)

- Coincide con la somma

A.S.E.

5.26

Prodotto di Interi Relativi (C-2)

- **N = 5**

Il prodotto di due numeri in C-2 non torna per i numeri negativi

Base 10	Base 2 (C-2)
14 x	01110
-13 =	10011
42 .	01110
14 .	011100
-182 .	0000000
	00000000
	01110000
	0100001010

$$(2^N + W) \cdot (2^N + Z) = 2^{2N} + 2^N (W + Z) + W \cdot Z$$

A.S.E.

5.27

Prodotto di Interi Relativi (C-2)

- **N=5 Estensione a 10**

Il prodotto di due numeri in C-2 torna se si estende la rappresentazione a 2N bit

Base 10	Base 2 (C-2)
-13 x	1111110011
14 =	0000001110
42 .	0000000000
14 .	1111100110
-182 .	1111001100
	1110011000
	0000000000
	1101001010

$$W \cdot (2^{2N} + Z) = \left| 2^{2N} \cdot W + W \cdot Z \right|_{2^{2N}} = \left| 2^{2N} + W \cdot Z \right|_{2^{2N}}$$

$$(2^{2N} + W) \cdot (2^{2N} + Z) = \left| 2^{4N} + 2^{2N} (W + Z) + W \cdot Z \right|_{2^{2N}} = \left| 2^{2N} + W \cdot Z \right|_{2^{2N}}$$

A.S.E.

5.28

Errori di rappresentazione 1

- In generale, la rappresentazione con un numero *finito* di cifre di un numero reale introduce errore
- Se lavoriamo con interi, possiamo convertire un numero decimale attraverso l'arrotondamento o il troncamento

11.6 12 (arrotondato) => 1100
 11 (troncato) => 1011

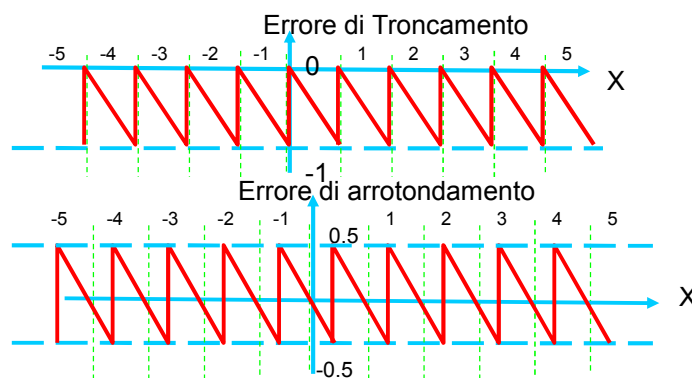
- Per numeri frazionari si procede in maniera analoga
 - 11.6531 su 2 cifre decimali → 11.65 (sia arr. che tronc.)

A.S.E.

5.29

Errori di rappresentazione 2

- Per i numeri negativi si applica la stessa definizione (si tronca verso $-\infty$)
- Va bene anche in complemento a 2



A.S.E.

5.30

Errori di rappresentazione 3

- Detta x^* la rappresentazione di x , si definiscono due errori:
 - Errore *assoluto*: $\varepsilon_A = |x - x^*|$
 - Errore *relativo*: $\varepsilon_R = |x - x^*|/|x|$
- Supponiamo di operare con 4 cifre decimale ($x^* = \pi \cdot 100$)
 - $314.1592 \rightarrow \varepsilon_A = 6.54 \cdot 10^{-5}, \varepsilon_R = 2.08 \cdot 10^{-7}$
- ($x^* = \pi$)
 - $3.1416 \rightarrow \varepsilon_A = 9.265 \cdot 10^{-5}, \varepsilon_R = 2.949 \cdot 10^{-5}$
- Inoltre, supponiamo di voler rappresentare distanze per uso scientifico:
 - Atomi: 10^{-10}m
 - Galassie: 10^{21}m

A.S.E.

5.31

Virgola mobile 1

- Dato un generico numero reale W
- Si può rappresentare in complemento a due con N bit utilizzando W^* tale che:

$$-2^{N-1} \leq W^* = \left[W \cdot 2^{Ex} \right] < 2^{N-1}$$

- Esempio $N = 10$ (+ esponente 6 bit)

$$W = 2 \times 2^0 = 2.0000 = 0000000010 \quad 000000$$

$$W = 43 \times 2^{-4} = 2.6875 = 0000101011 \quad 111100$$

$$W = 347 \times 2^{-7} = 2.7109 = 0101011011 \quad 111001$$

mantissa

esponente

A.S.E.

5.32

Virgola mobile 2

- **Fra tutte le rappresentazioni possibili ne esiste una che utilizza al meglio la dinamica a disposizione (minimo errore)**

$$W = 347 \times 2^{-7} = 2.7109 = 0101011011 \quad 111001$$

- **Per numeri positivi è**

$$2^{N-2} \leq W^* < 2^{N-1} \quad (\text{per } N=10 \quad 256 \leq W^* \leq 511)$$

- **Per numeri negativi è**

$$-2^{N-1} \leq W^* < -2^{N-2} \quad (\text{per } N = 10 \quad -512 \leq W^* \leq -257)$$

A.S.E.

5.33

Virgola mobile 3

- **La rappresentazione normalizzata è caratterizzata dal fatto che le due cifre più significative sono diverse**
- **Esempio rappresentare π su 10 bit**

$$\text{dinamica per } N=10 \quad -512 \leq Z \leq 511$$

$$k = \frac{511}{\pi} = 162.65 \quad (128 < k < 256)$$

$$E_0 = -7$$

Rappresentazione normalizzata

$$\text{Int}[\pi \cdot 2^7] = 402 = 0110010010 \cdot 2^{-7}$$

A.S.E.

5.34

Virgola mobile 4

- **Esempio rappresentare $-\pi$ su 10 bit**

dinamica per $N = 10$ $-512 \leq Z \leq 511$

$$k = \frac{512}{\pi} = 162.9 \quad (128 < k < 256)$$

$$E_0 = -7$$

Rappresentazione normalizzata

$$\text{Int}|\pi \cdot 2^7| = 402 = 0110010010 \cdot 2^{-7}$$

$$-\pi = 1001101110 \cdot 2^{-7}$$

A.S.E.

5.35

Aritmetica in Virgola Mobile

- Consideriamo $X = X_M \times B^{X_E}$ ($Y_E > X_E$)
- Somma $Z = X + Y = (X_M 2^{X_E - Y_E} + Y_M) \times 2^{Y_E}$
- Sottrazione $Z = X - Y = (X_M 2^{X_E - Y_E} - Y_M) \times 2^{Y_E}$
- Moltiplicazione $Z = X \times Y = (X_M \times Y_M) \times 2^{X_E + Y_E}$
- Divisione $Z = X \div Y = (X_M \div Y_M) \times 2^{X_E - Y_E}$
- Somma e sottrazione sono più complesse di moltiplicazione e divisione!
 - Occorre allineare gli esponenti prima di effettuare l'operazione

A.S.E.

5.36

Somma in virgola mobile

- calcolare $\pi + \ln 59 = 3.1415 + 4.0775 = 5.219$

$$\pi = 0110010010 \cdot 2^{-7}$$

$$512 / \ln 59 = 125.5 \quad E_0 = -6$$

$$\ln 59 = 260 \cdot 2^{-6} = 0100000100 \cdot 2^{-6}$$

$$\pi + \ln 59 = (0100000100 + 0011001001) \cdot 2^{-6} =$$

$$0111001101 \cdot 2^{-6} = 461 \cdot 2^{-6} = 7.2031$$

A.S.E.

5.37

Prodotto in virgola mobile

- calcolare $\pi \cdot \ln 59 = 3.1415 \cdot 4.0775 = 12.8099$

$$\pi = 0110010010 \cdot 2^{-7}$$

$$\ln 59 = 0100000100 \cdot 2^{-6}$$

$$\pi \cdot \ln 59 = (0100000100 \cdot 0110010010) \cdot 2^{-(7+6)} =$$

$$= 0110010110 \cdot 2^8 \cdot 2^{-13} = 406 \cdot 2^{-5} = 12.6875$$

A.S.E.

5.38

Errore

- Nella rappresentazione in virgola fissa l'errore è assoluto (± 0.5)
- Nella rappresentazione in virgola mobile l'errore è relativo

A.S.E.

5.39

Osservazione 1

- La mantissa di numeri positivi normalizzati è del tipo **01XXXXXXXX**
 - Esempio su 8 bit si ha
 - $01111111 > W > 01000000$
 - $127 > W > 64$
- La mantissa di numeri negativi normalizzati è del tipo **10XXXXXXXX**
 - Esempio su 8 bit si ha
 - $10000000 > W > 10111111$
 - $-128 > W > -65$

A.S.E.

5.40

Osservazione 2

- I numeri normalizzati possono essere interpretati come sola parte frazionaria, cioè
- Si cambia l'esponente e si divide la mantissa per il massimo modulo rappresentabile
 - Esempio su 8 bit
 - il max modulo rappresentabile è 128
 - Per numeri positivi si ha
 - $01111111 > W > 01000000$
 - $0.9922 > W > 0.5$
 - Per numeri negativi si ha
 - $10000000 > W > 11111111$
 - $-1.00 > W > -0.508$

A.S.E.

5.41

Osservazione 3

- La precedente rappresentazione può essere utilizzata moltiplicando la mantissa per 2
- Quindi si ha
- Per i numeri positivi la rappresentazione è
 - $1.984 > w > 1$
- Per i numeri negativi la rappresentazione è
 - $-2.00 > W > -1.016$

A.S.E.

5.42

Osservazione 4

- I numeri positivi sono ancora rappresentati nella forma
– 01XXXXXX
- I numeri negativi sono ancora rappresentati nella forma
– 10XXXXXX
- La prima cifra dopo il bit di segno è sempre nota, quindi non c'è informazione e può essere omessa
- Ovvero non si trasmette l'uno prima del punto
- Si ha così un bit in più per la rappresentazione della mantissa
- Ovvero si dimezza l'errore

A.S.E.

5.43

Conclusioni

- Codici numerici
- Codici alfanumerici
- **RIEPILOGO**
- Virgola mobile

A.S.E.

5.44